# REA: The real estate agent

ein Projekt der „Gesture and Narrative Language Group“
des MIT Media Laboratory


Ausarbeitung eines Referates von Tim Scheele

## 1. Anforderungen an einen Embodied Conversational Agent

Rea wurde in der Arbeitsgruppe von Justine Cassell am MIT als virtuelle Immobilienmaklerin entwickelt. Dieser Agent soll ein möglichst natürlicher Kommunikationspartner sein. Im Idealfall soll so ein Benutzer ein Computersystem bedienen können, ohne sich vorher in die Bedienung einzuarbeiten. Desweiteren ermöglicht eine multimodale Ausgabe einen solideren Informationsaustausch.
Deshalb wurden von dem Rea-Projekt folgende, für eine Konversation grundlegenden Verhaltensweisen gefordert. Die Reaktion auf verbale und nonverbale Eingaben, genau wie die Generierung verbaler und nonverbaler Ausgaben. Das heisst es soll eine Unterhaltung eines Benutzers mit Rea möglich sein. Hinzu kommen konversationale Funktionen (siehe Ausarbeitung Termin 2), also turn-taking und –giving zu erkennen und entsprechende Reaktionen einzuleiten. Schließlich wird von einem natürlichen Kommunikationspartner noch erwartet, daß er neue Themen in die Unterhaltung einbringt, also Eigeninitiative zu zeigen in der Lage ist.
Diese Punkte sind zumindest grundsätzlich verwirklicht worden, auch wenn die Natürlichkeit nur Ansatzweise vorhanden ist. Rea spricht den Benutzer an, wenn er ihren Arbeitskreis betritt, gibt den turn ab, wenn er beginnt zu sprechen, oder wenn sie eine Aufgabe erfüllt hat und nimmt ihn, wenn der Benutzer eine längere Pause macht.

## 2. Die Architektur von Rea

Die Verarbeitung einer Benutzereingabe ist in verschiedene Arbeitschritte und damit in verschiedene Module aufgeteilt (siehe Abb.1). Der Input-Manager sortiert die Benutzereingaben zeitlich, dass Understanding Module ordnet diesen Eingaben Funktionen zu, auf welche das Decision Module entsprechende Reaktionen plant. Diese werden von dem Generation Module in auszuführende Verhaltensweisen übersetzt und schließlich vom Action Scheduler realisiert. Die gesamte Kommunikation zwischen den Modulen geschieht mit Hilfe von KQML-frames. Diese beinhalten zwischen den jeweils äusseren Komponenten, also zwischen dem Input Manager und dem Understanding Module, sowie dem Generation Module und dem Action Scheduler noch Beschreibungen von Verhaltensweisen, während sie in der internen Kommunikation die Funktionen von Verhalten übertragen.

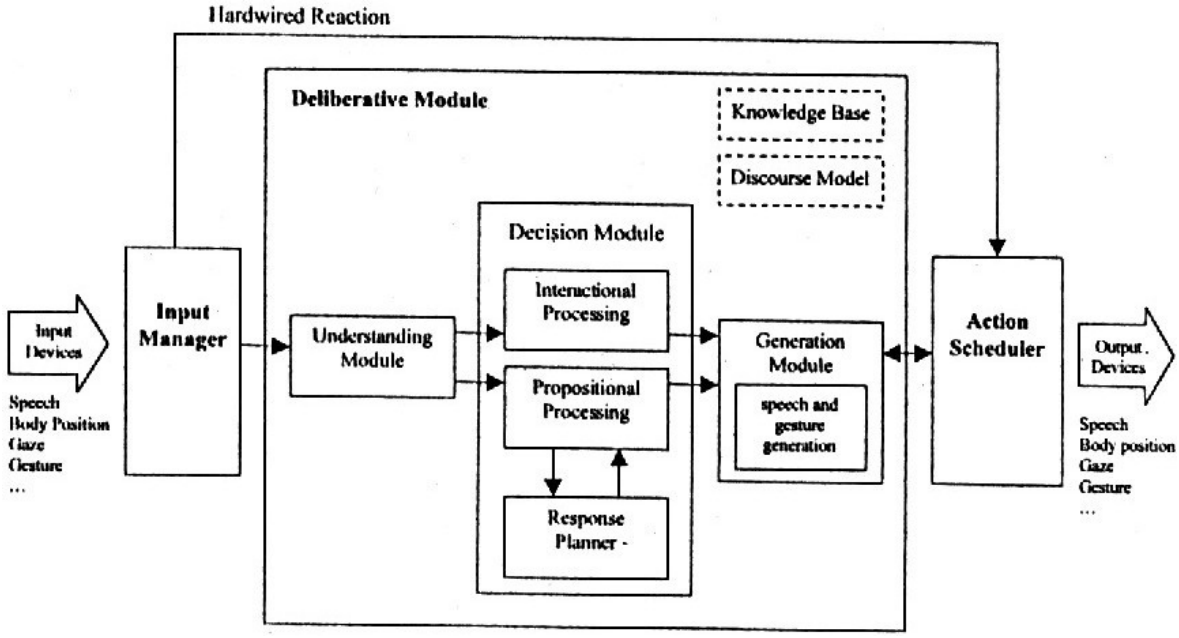


Abb.1 Die Architektur von Rea

### 2.1. Die Eingabemöglichkeiten und der Input-Manager

Zwei über der Leinwand montierte Kameras erfassen die Position und die Ausrichtung der Hände und des Kopfes des Benutzers. Dieser trägt ein Mikrophon, welches einerseits zur Erkennung des Anfangs und des Endes einer sprachlichen Äußerung eingesetzt wird, anderseits mittels IBMs ViaVoice98 Texthyphothesen aus der Sprache generiert.
Die Bildinformation wird vom *Input Manager* momentan nur benutzt, um festzustellen ob ein Benutzer sich im Aktionsraum befindet und wenn ja, wo er ist. Diese Informationen werden zusammen mit der Information ob der Benutzer spricht oder nicht, fest verdrahtet an den Action Scheduler geschickt, welcher instantan entsprechende Aktionen einleitet, wie z. B. den Benutzer mit den Augen zu verfolgen. Ausserdem werden diese Daten noch zwischengespeichert, um im zeitlich korrekten Rahmen zusammen mit den Texthypothesen an das Understanding Module weitergegeben zu werden.

## 2.2. Das Understanding Module

Das *Understanding Module* hat die Aufgabe den Benutzereingaben bekannte Funktionen zuzuordnen. In erster Linie heißt das, daß den verschiedenen Texthypothesen nun Einträgen der Wissensbasis zugeordnet werden. Des weiteren werden sowohl die aktuellen, als auch die Informationen zu dem vorangegangenen Zeitschritt über die Präsenz des Benutzers und seinem Sprachzustand verwendet, um einer bestimmten Eingabe eine Diskursfunktion zuzuordnen. Die resultierenden Funktionen werden an das Decision Module übergeben.

Table 1. stellt die möglichen Diskursfunktionen dar, welche einer Benutzereingabe in zwei möglichen Zuständen im Understanding Module zugeordnet werden.

| State | User Input | Input Function |
|---|---|---|
| Rea speaking | Gesture | Wanting turn |
| | Speech | Taking turn |
| User speaking | Pause of <500 msec. | Wanting feedback |
| | Imperative phrase | Giving turn |
| | Interrogative phrase | Giving turn |
| | Declarative phrase & pause >500 msec. & no gesture | Giving turn |
| | Declarative phrase & long gesture or pause | Holding turn |

*Table 1. Functional interpretation of turn taking input*

## 2.3. Das Decision Module

Das *Decision Module* ist in zwei verschiedene Bereiche aufgeteilt, die interaktionale und die propositionale Berechnung. In der interaktionalen Berechnung wird der folgende Zustand des Diskurses ermittelt. Dies geschieht wiederum auf der Grundlage des alten Zustandes und einer möglichen Diskursfunktion von der Seite des Benutzers.

Der nebenstehende Automat stellt die interaktionalen konversationalen Zustände und deren mögliche Übergänge ineinander dar. Nur der Benutzer ist hier in der Lage, „On Hold“ zu sein, sprich nichts zu tun und trotzdem den Turn zu behalten.

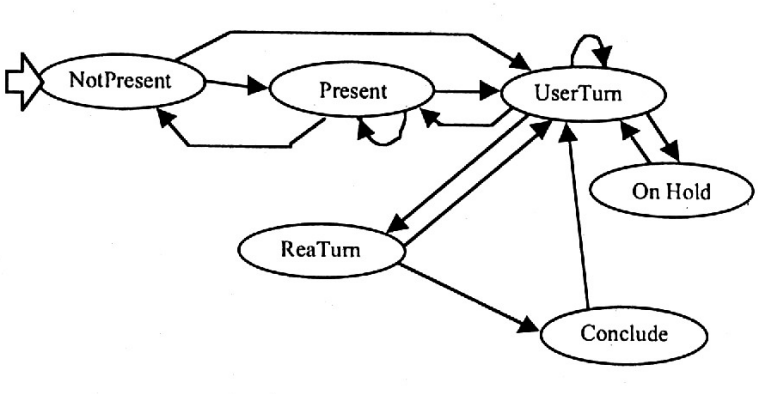


Abb.2 mögliche Zustände im Rea-System

Die propositionale Berechnung umfaßt die Sprache und unterstützende Gestik. Die Informationen aus dem Sprechakt des Benutzers werden mit der dynamischen Wissensbasis abgeglichen. In dieser stehen alle wichtigen Informationen, welche im Verlauf des Gespräches schon Teil der Unterhaltung waren, z. B. ob ein bestimmtes Objekt schon eingeführt wurde und somit ein

Bezug darauf hergestellt werden kann. Des weiteren wird untersucht, ob der Agent eine bestimmte Anfrage oder Aufgabe erhalten hat, welche er bearbeiten muß. Ist dies der Fall, so wird weiter untersucht, wie komplex diese Aufgabe ist und, wenn sie mehrere Aktionen von Rea umfasst, an den Response Planner übergeben. Dieser spaltet die Aufgabe in mehrere kleine Einzelaufgaben, welche er dann nach und nach mit etwaigen interaktionalen Funktionen an das Generation Module weiterreicht.

## 2.4. Das Generation Module

Im *Generation Module* wird diese Funktion in angemessene Verhaltensweisen übersetzt. Eine Hauptkomponente dieses Modules ist der SPUD-Server (siehe Abschnitt 3). Die eintreffenden Diskursfunktionen werden in kommunikative Ziele, welche der SPUD-Server verstehen kann, umgewandelt. Dieser erzeugt dann wiederum auf der Grundlage des konversationalen Zustandes und der dynamischen Wissensbasis eine natürlichsprachliche Aussage und gegebenenfalls unterstützende oder zusätzliche Gestik, z. B. Augenbrauen hochziehen.

## 2.5. Der Action Scheduler

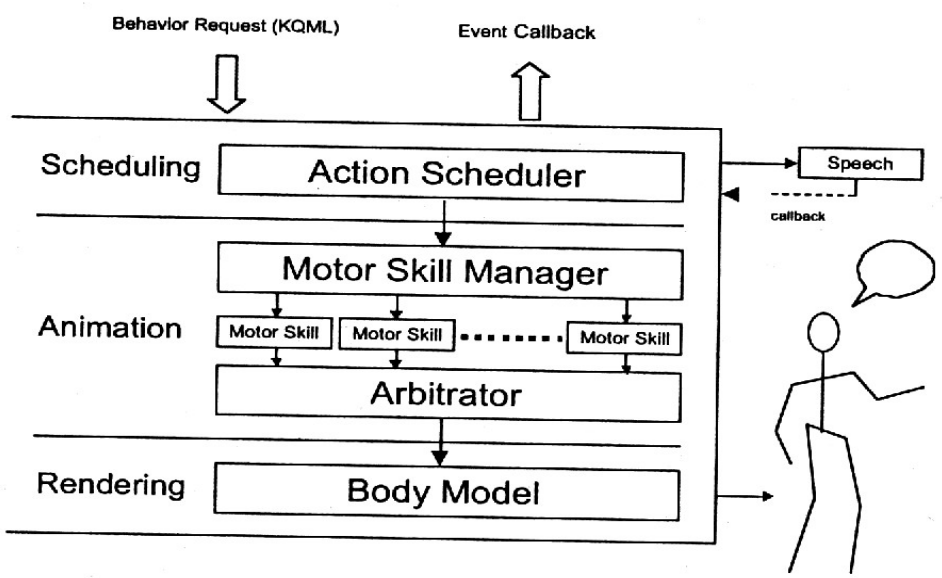


Abb. 3 Die Verarbeitung von Verhaltensfunktionen zu Aktionen

Der *Action Scheduler* ist für die Umsetzung der Verhaltensbeschreibungen in Bild und Ton zuständig. Er ist wiederum in 3 Untereinheiten aufgeteilt (siehe Abb. 3) . Diese verarbeiten allerdings nur die Bildinformationen, die Sprache wird an einen externen Spracherzeuger geschickt, welcher den zu sprechenden Text in einem Wave-file zurückgibt. Der Action Scheduler plant nun, zu welchem Zeitpunkt welche Aktion ausgeführt werden soll. Hierbei muss er entscheiden, welche Aktionen die höhere Priorität haben. Die hartverdrahtete Aktion, dass Rea den Benutzer immer mit den Augen verfolgt, wird zum Beispiel von der Aktion nach hinten zu blicken ersetzt, da diese höhere Priorität besitzt.

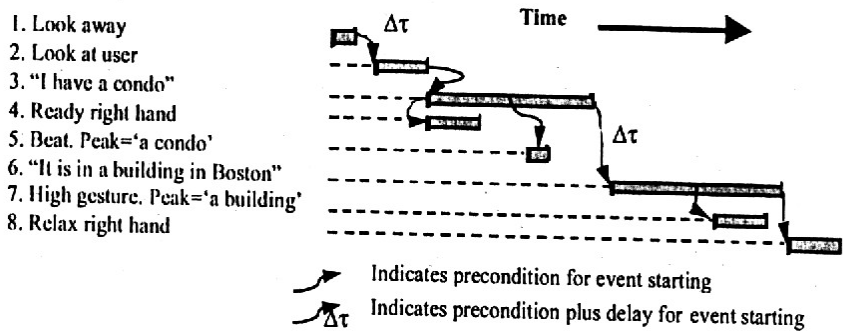


Abb. 4 Die Event getriebene Verarbeitung einzelner Aktionen

Die zeitliche Synchronisierung erfolgt Event getrieben. Dies liegt unter anderem daran, dass der Sprachsynthesizer keine zeitlichen Informationen, also Phonem- oder Wortlängen, zurückgibt. Deswegen kann man bedeutungsvolle Ereignisse markieren, z.B. Anfang oder Ende jedes Wortes, und so bestimmte Aktionen synchron zu einem bedeutungstragenden Wort ausführen, z. B. unterstützende Beat-Gesten, oder ein Zeigen auf einen Teil eines Hauses.
In Abb. 4 ist ein Beispiel für dieses Verfahren. Die Pfeile deuten immer an, das hier ein Event vorliegt, welches den Beginn einer neuen Aktion auslöst. Die mit $\Delta\tau$ beschrifteten Pfeile weisen auf eine zeitliche Verzögerung hin, die neue Aktion beginnt $\Delta\tau$ nach dem bezeichneten Event.
Abgesehen von der Sprache resultieren alle anderen Verhaltensweisen Reas schließlich in Bewegungen des Körpers. Der Motor Skill Manager ist in mehrere einzelne Motor Skills unterteilt. Ein Motor Skill stellt eine bestimmte Art der Animationsberechnung dar. Die Motor Skills gehen von einfachen straight-forward Berechnung, wie ein Kopfnicken, zu inverser Kinematik und Key-frame Animationen. Die darzustellende Verhaltensweise wird also in die einzelnen Animationsarten unterteilt. Der Schiedsrichter (Arbitrator) hat dann Zugriff auf dass Körpermodell. Dieses ist auf der Grundlage der H-Anim VRML-Spezifikation modelliert. Dies beinhaltet unter anderem, dass der Körper in verschiedene Freiheitsgrade aufgeteilt ist, z. B. die einzelnen Gelenke des Skeletts oder die Augen. Jedes Motor Skill, welches einen bestimmten Teil des Körpers animieren soll, fragt nun bei dem Schiedsrichter an, ob dieser Freiheitsgrad noch frei ist. Im Falle einer Mehrfachbelegung wird wieder verglichen, welche Animation die höhere Priorität hat. Diese wird dann ausgeführt, während die andere zurückgestellt oder gelöscht wird. Schliesslich überwacht der Schiedsrichter die Ausführung der einzelnen Aufträge. Das Körpermodell wird gerendert und synchron mit der Sprachausgabe auf die Leinwand projeziert.

## 2.6. Ein Beispiel

Das folgende Beispiel zeigt, welche Funktionen Rea in einer Konversation ausführen kann. Anfangs reagiert sie darauf, das jemand ihren Arbeitsraum betritt und spricht ihn an. Im weiteren Diskurs führt sie neue Objekte in die Unterhaltung ein, kann darauf referenzieren, benutzt unterstützende Gestik und reagiert schliesslich auf Zeichen, welche der Benutzer (Lee) macht, um den turn zu bekommen.

*Lee approaches the projection screen. Rea is currently turned side on and is gazing idly about. As Lee moves within range of the two cameras mounted above the screen, Rea turns to face him and says:*
REA> Hello. How can I help you?
LEE> I´m looking to buy a place near MIT.
Rea nod, indicating that she is following.
REA> I have a house to show you.
*A picture of a house appears on-screen behind Rea who blinks and looks at the house and then at Lee.*
REA> It´s in Somerville.
LEE> Tell me about it.
*Rea looks up and away while she plans what to say.*
REA> It´s big.
*Rea makes an expansive gesture with her hands.*
*Lee brings his hand up as if he is about to speak and so Rea does not continue; instead, she is waiting for him to speak.*
LEE> Tell me more about it.
REA> Sure thing. It has a nice garden.
*Rea sketches a curve gesture with her hands indicating that the garden extends along two sides of the house.*
LEE> How far is it?
REA> It is five minutes to the Porter Square T Station.
*Rea makes it clear that it is five minutes **on foot** from the T Station by making a walking gesture with her fingers.*
LEE> How big is the house?
REA> It has four bedrooms, three bathrooms...
Lee interrupts Rea who stops speaking immediately.
LEE> Wait. Tell me, where is the master bedroom?
REA> I´m sorry, I didn´t catch that. What did you ask me?
LEE> Where is the master bedroom?
REA> It´s upstairs.
*Rea points up.*
LEE> Where is the master bathroom?
REA> It´s next to the bedroom.
*Rea brings her hands together to indicate the relationship between the bedroom and the bathroom.*

## 3. Sentence Planner Using Descriptions (SPUD)

Matthew Stone und Christine Doran haben den SPUD Algorithmus entwickelt, um aus einem gegebenen Ziel, der Beschreibung eines Ereignisses oder Zustandes, einen natürlichsprachlichen Satz zu erzeugen, welcher dieses Ziel erfüllt. Dazu wird eine LTAG (Lexicalized Tree Adjoining Grammar) verwendet, eine Grammatik, welche in einer Baumstruktur Syntax und Semantik der einzelnen Wörter und möglicher Sätze definiert und durch Ersetzung einen Satz erzeugt. Als weitere Grundlage dienen die Zustände des Dikurses. Aus ihnen wird zum Beispiel ersichtlich, ob ein bestimmtes Objekt schon bekannt ist, oder erst noch eingeführt werden muss.

### 3.1. LTAG

Eine TAG besteht aus elementaren Bäumen, welche nicht mehr unterteilt werden können, und Regeln zu deren Erweiterung. Dies sind in diesem Fall Ersetzung (Sustitution) und Angrenzung (Adjoining). In den Beschreibungen der Elementarbäume ist unter anderem enthalten, in welchem Kontext eines Diskurses sie benutzt werden können und sollen. Das beinhaltet auch die Information ob und auf welche Weise sie erweitert werden können. Die Ersetzung kann nur an den Blättern eines Baumes geschehen, also an Endpunkten. Dieses Blatt wird durch einen Elementarbaum eines bestimmten Formates vollständig ersetzt. Die Angrenzung kann auch an einem Knoten in der Mitte des Baumes vorgenommen werden, da hier ein Elementarbaum eingefügt wird. Die weitere Verästelung des Baumes zu den Blättern hin wird an der entsprechenden Stelle des Elementarbaumes wieder angesetzt.
In SPUD wird die Semantik der Bäume durch die Anwendung von zwei Prinzipien auf den LTAG Formalismus spezifiziert. Zum einen gibt es mehrere mögliche Typen von Bäumen. Die Bedeutung eines Baumes ergibt sich durch die Möglichkeiten seine Blätter zu ersetzen. Zum anderen wird jeder syntaktische Knoten beschriftet und kann nur von einem Knoten desselben Typs ersetzt werden. Diese syntaktischen Parameter werden von einer Wissensbasis instantiiert.

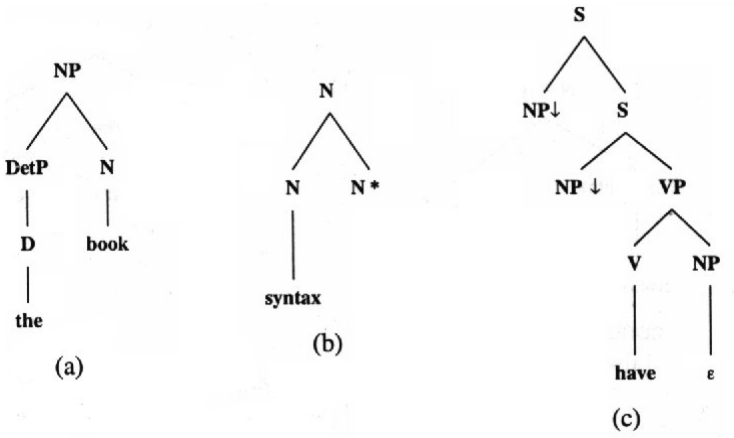


Abb. 5. Einige LTAG Bäume. (a) NP, (b) Noun-Noun Compound, (c) Topicalized Transitive Der Stern in (b) zeigt an, das der Baum hier durch Angrenzung erweitert werden kann, der Pfeil in (c) zeigt eine mögliche Erweiterung durch Ersetzung

## 3.2. Die Grundlagen der SPUD

In jedem Elementarbaum ist spezifiziert, unter welchen kontextuellen Bedingungen er in dem Diskurs benutzt werden kann. Dies wird in vier verschiedenen Pragmatiken festgelegt. Die Objekte unterscheiden sich in ihrer Neuheit im Bezug auf den Hörer und den Diskurs (Ist dieses Objekt schon Bestandteil der Unterhaltung gewesen?). Zweitens haben verschiedene Objekte verschiedene Prominenz (salience), welche angibt, wie zugänglich dieses in dem momentanen Kontext ist. Drittens wird die Beziehung eines Objektes zu anderen Objekten spezifiziert. Dies beinhaltet z. B. ob ein Objekt Teil eines anderen ist, oder ob es in der gleichen Klasse ist. Schliesslich kann der Diskurs auch einige offene Probleme (open propositions) beinhalten, welche bestimmte Typen von Objekten bevorzugt.
Ein NP-Baum zum Beispiel beinhaltet einen Bezeichner (determiner) und ein Hauptwort. Je nachdem, ob das Hauptwort im Diskurs genau zu identifizieren ist oder nicht, wird für den Bezeichner „the“ oder „a“ , „an“ eingesetzt.
Ausserdem gibt es sogenannte S-Bäume, welche die Satzart festlegen. Grundsätzlich ist dies die Subjekt-Verb-Objekt-Form, oder eine Variante davon.
Das System kann zwei Arten von Zielen verfolgen. Erstens ein Ziel der Form „zeichne x als Katze aus“. In diesem Fall muss eine Beschreibung des Objektes „x“ angefertigt werden, wobei die syntaktische Kategorie Katze verwendet wird. Zweitens die Anweisung über „p“ zu reden. Das Problem „p“ wird dem Algorithmus hinzugefügt und muss unter Verwendung des geteilten Wissens gelöst werden.

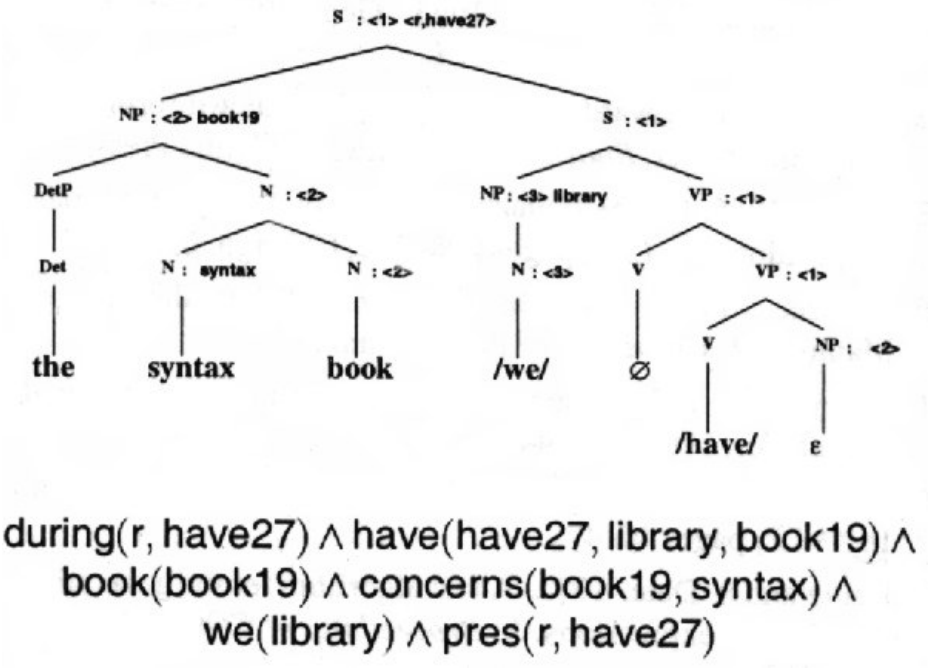


Abb. 6. Eine komplette Antwort des SPUD-Systems. Es übernimmt hier die Rolle einer Bibliotheksauskunft, welcher auf der Basis der Bibliotheksdatenbank Auskünfte über den Leihstatus bestimmter Bücher geben kann.

## 3.3. Der Algorithmus

Bei jeder Iteration des Algorithmuses wird die geeignetste Erweiterung des Baumes ausgesucht. Nun werden passende lexikalische Einträge ausgesucht. Diese Einträge müssen natürlich an die gewünschte Stelle des Satzes passen und die gewünschte Information liefern. Im nächsten Schritt wird der passende Elementarbaum ausgewählt (Ein Eintrag im Lexikon kann mit mehreren Baumstrukturen verbunden werden). Schliesslich werden alle Vorschläge unter Verwendung der Pragmatiken verglichen und der beste wird in den Hauptbaum eingesetzt. Mögliche neue Ziele, die durch diese Ersetzung aufgekommen sind, werden der Liste der zu erfüllenden Ziele hinzugefügt. Der Algorithmus wiederholt diese Prozedur so lange, bis alle Ziele erfüllt sind.

Im Rea-System wurde der SPUD Algorithmus erweitert, in dem noch die Verwendung von Gestik hinzugefügt wurde. Dies wurde erreicht, indem einzelne Gesten genau wie Wörter in das Lexikon aufgenommen wurden. Der Algorithmus kann also entscheiden, ob er ein Wort oder eine Geste verwendet, um ein bestimmtes Ziel zu erreichen.

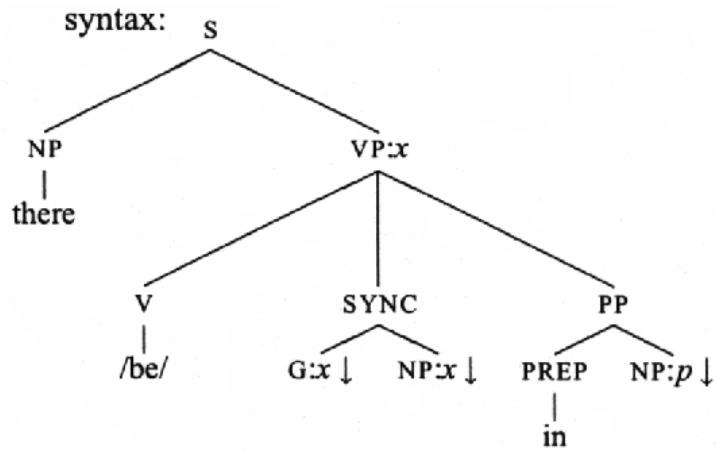


Abb. 7 zeigt die gleichberechtigte Verwendung von Gestik

## 4. Ausblick

Durch einige Probleme in dem Rea Projekt, ist der Anspruch eine natürliche Konversation zu führen noch nicht erfüllt, welche damit die weitere Arbeit an Rea festlegen. Zum einen ist es die Tatsache, dass noch nicht bekannt ist, wie eine konversationale Funktion ein entsprechendes Verhalten erzeugt. Momentan können nur festgelegte Verhaltensweisen zu entsprechenden Funktionen ausgeführt werden. Das Ziel ist hier, zu verstehen, wie eine Funktion ein bestimmtes Verhalten generiert und dieses zu formalisieren.
Das größere Problem, dessen Bearbeitung in näherer Zukunft liegt, ist das Problem der zeitlichen Differenzen. Einerseits hat Rea schnelle Reaktionen, nämlich die fest verdrahteten, zum anderen erfordert die Planung eines Satzes so viel Zeit, dass einige Benutzer den Eindruck hatten, Rea sei ein bisschen „benebelt“. Dies erfordert entweder mehr Rechenleistung oder einen schelleren Satzplaner.
Die schlechte Synchronisierung soll durch ein neues Text-to-Speech System gelöst werden, welches auch die Länge jedes einzelnen Phonems bereitstellt. Ausserdem soll noch der Eindruck getestet werden, den Rea auf neue „untrainierte“ Benutzer hat. Bis jetzt hat sich nur gezeigt, dass die unterstützende Gestik zu einem natürlicheren Spracheindruck beiträgt.

## 5. Quellen und Literatur

[Cassell et al. 2000] J. Cassell, J.Sullivan, S. Prevost, E. Churchill, *Embodied Conversational Agents, Kap. 2.*
The MIT Press, 2000.

[Cassell et al. 1999] J. Cassell, T. Bickmore, M. Billinghurst, L Campbell, K. Chang, H. Vilhjálmson, H. Yan, *Embodiement in Conversational Interfaces: Rea.*
Association for Computing Machinery, Inc, 1999.

[Stone, Doran 1997] M. Stone, C. Doran, *Sentence Planning as Description Using Tree Adjoining Grammar.*
Proceedings of ACL 1997, pages 198--205.

Web-Links:

Die Arbeitsgruppe von Justine Cassell:
http://gn.www.media.mit.edu/groups/gn/index.html

Die Homepage von Matthew Stone (SPUD):
http://www.cs.rutgers.edu/~mdstone/nlg.html

Die Homepage der Sprache CLIPS:
http://www.ghg.net/clips/CLIPS.html

Infos zu KQML:
http://www.cs.umbc.edu/kqml/

Die H-Anim VRML Spezifikation:
http://ece.uwaterloo.ca/~h-anim/newspec.html